**Noname manuscript No.**
(will be inserted by the editor)

# Analyzing the Exhaustiveness of the Synapse Protocol

**Bojan Marinković · Vincenzo Ciancaglini · Zoran Ognjanović · Paola Glavan · Luigi Liquori · Petar Maksimović**

the date of receipt and acceptance should be inserted later

**Abstract** The Synapse protocol is a scalable protocol designed for information retrieval over inter-connected heterogeneous overlay networks. In this paper, we give a formal description of Synapse using the Abstract State Machines framework. The formal description pertains to Synapse actions that manipulate distributed keys. Based on this formal description, we present results concerning the expected exhaustiveness for a number of scenarios and systems maintained by the Synapse protocol, and provide comparisons to the results of the corresponding simulations and experiments. We show that the predicted theoretical results match the obtained experimental results, and give recommendations on the design of systems using Synapse.

**Keywords** Peer-to-peer · DHT-based overlay networks · Abstract State Machines · Retrieval probability

## 1 Introduction

Overlay networks have recently been identified as a promising model that could cope with the current issues of the Internet, such as scalability, resource discovery, failure recovery, routing efficiency, and, in particular, in the context of information retrieval. Today, one can notice that not only many disparate overlay networks co-exist across the Internet, but that there are scenarios in which they also compete for the same resources on shared nodes and underlying network links.

One of the main problems of overlay networking is how to allow different overlay networks to interact and co-operate with each other. When it comes to the overlay networks that have already been developed, one can perceive a great extent of heterogeneity between them. In most cases, this heterogeneity renders them unable to co-operate, communicate, and exchange resources with one another without resorting to the costly, non-scalable, and security-compromising operation that is overlay merging.

On the other hand, there are many situations where different overlay networks could benefit from co-operation for various purposes, such as collective performance enhancement, larger shared information, better resistance to loss of connectivity (network partitions), improved routing performance in terms of delay, throughput, and packets loss (by, for instance, co-operative forwarding of flows). In the context of large-scale information retrieval, several overlays may wish to offer an aggregation of their resources to their potential common users, without relinquishing control over them.

In terms of fault-tolerance, co-operation can increase the availability of the system – if one overlay were to become unavailable, the global network would only undergo partial failure, as other different resources would still be usable. A solution could be found in using a meta-protocol that allows a request to be routed through multiple heterogeneous overlay networks, thus increasing the success rate of every request.

The ready-to-market Distributed Hash Tables (DHT) - based technology of structured overlay networks is enriched with the new capability of crossing different overlays through co-located nodes, i.e. by peers who are, by user's choice,

The work presented in this paper was supported by the Serbian Ministry of Education, Science and Technological Development, projects ON174026 and III44006, through Matematički Institut SANU and by Ministarstvo znanosti, obrazovanja i športa republike Hrvatske.

Bojan Marinković · Zoran Ognjanović · Petar Maksimović
Mathematical Institute of the Serbian Academy of Sciences and Arts - MISANU, Serbia E-mail: bojanm@mi.sanu.ac.rs

Vincenzo Ciancaglini · Luigi Liquori · Petar Maksimović
National Institute for Research in Computer Science and Control - INRIA, France

Paola Glavan
Faculty of Mechanical Engineering and Naval Architecture - FSB, Croatia

member of several overlays. Such nodes are themselves not only able to query multiple overlays in order to find a match, but can also replicate requests, passing them through from one network to another, and collecting the multiple results.

One of the possible solutions for the inter-connection of heterogeneous overlay networks is the Synapse protocol, introduced in [2], [5] and [8]. It is a generic and flexible meta-protocol that provides simple mechanisms and algorithms for easy interconnection of overlay networks. The first contribution of this paper, motivated by the ever-growing need for formal correctness, will be the formal specification of Synapse within the formalism of Abstract State Machines.

Abstract State Machines (ASM), introduced in [1], [3] and [4], are versatile machines which are able to simulate arbitrary algorithms (including programming languages, architectures, distributed and real-time protocols, etc.) in a direct and essentially coding-free way. The simulator is not supposed to implement the algorithm on a lower abstraction level; the simulation should be performed on the natural abstraction level of the algorithm. A vast literature on ASMs shows how to model closely and faithfully real complex systems and how to use models in order to verify their properties. Some well known algorithms, like Bakery algorithm, Rail road crossing problem, Kerberos algorithm, Java Virtual Machine, etc., were described using ASM.

In addition to the specification of Synapse in ASM, we will provide a probabilistic estimate on the exhaustiveness of the Synapse protocol across a number of scenarios. To summarize, in this paper we aim to:

- Give a specification of the Synapse protocol using the formalism of ASM,
- Theoretically analyze the exhaustiveness of the Synapse protocol, and
- Describe and run the corresponding experiments, in order to validate the obtained theoretical results.

In doing so, we provide a starting block from which further formal analysis of the Synapse protocol can be performed, as well as an easy mechanism for estimation of the exhaustiveness of the Synapse protocol, justified by the performed experiments and simulations.

### 1.1 Outline of the paper

The rest of paper is organized as follows. Section 2 contains the basic introduction to the Synapse protocol. In Section 3, we describe the Synapse protocol using the formalism of ASM. In Section 4, we present the expected results concerning exhaustiveness of the Synapse protocol and compare them to the results obtained through corresponding simulations and experiments. In Section 5, we introduce several possible improvements to the original protocol and compare the expected exhaustiveness of an improved version of Synapse with the original one. Finally, we present our conclusions and outline directions for further work in Section 6.

## 2 Introduction to the Synapse Protocol

The Synapse protocol is a scalable protocol for information retrieval over inter-connected (heterogeneous) overlay networks. Synapse is based on a set of co-located nodes, also called synapses, serving as low-cost natural candidates for intra-overlay bridges. In the simplest case, where overlays to be interconnected are ready to adapt their protocols to the requirements of inter-connection, every message received by a co-located node can be forwarded to other overlays to which that node belongs. In other words, the node, upon receipt of a search query, in addition to its forwarding to the next hop in the current overlay according to its routing policy, can potentially start a new search, according to some given strategy, in some or all of the other overlay networks it belongs to. This implies that a mechanism that limits the lifespan or lifetime of a query, Time-To-Live (TTL), needs to be provided and the detection of already processed queries implemented, so as to avoid infinite looping in the networks, as is the case in unstructured peer-to-peer systems. Applications of top of Synapse see the inter-connected overlays as an unique overlay.

The inter-overlay network induced by the Synapse protocol can be considered as an aggregation of heterogeneous sub-overlay networks (hereinafter referred to as intra-overlay networks). Each intra-overlay consists of an instance of, e.g., Chord or any structured, unstructured, or hybrid overlay, equipped with a $\langle key, value \rangle$ distribution and retrieval mechanism. We recall that an overlay network for information retrieval consists of a set of nodes on which information concerning certain resources is distributed. Each intra-overlay has its own logical topology, search complexity, as well as routing and fault-tolerance mechanisms.

As such, we have developed two models of the Synapse protocol; the first, *white box* model, is suitable for intra-connecting overlays whose standards are open and collaborative, meaning that the protocol and the software client can be modified accordingly. The second, *black box* model, is suitable for inter-connecting overlays that, for different reasons, are not collaborative at all, in the sense that they only route packets according to their proprietary and immutable protocol. The white box allows the adding of extra parameters to the current intra-overlay we are connecting, while the black box deals with those extra parameters by means of a synapse control network, i.e. a distributed overlay that stores all the synapse parameters that cannot be carried on by the overlay we are traversing.

The white box Synapse connects heterogeneous network topologies under the assumption that each node is aware of

the additions made to existing overlay protocols. The new parameters for handling the game-over strategy and replication are embedded into the existing protocols. One important requirement of the White box Synapse model w.r.t. other hash-based protocols is that the addresses of keys and nodes circulate unhashed from hop to hop, so that it can be rehashed once a synapse is encountered. Hash functions have no inverse: once a sought key is hashed, it is impossible to retrieve its initial value for rehashing during the forwarding of the request across overlays. Naturally, hash functions may vary (in implementations and keysize) from overlay to overlay. Both the hashed and the original key can be carried within the message, or a fast hash computation can be performed at each step. Standard cryptographic protocols can be used to protect the unhashed key in case of strong confidentiality requirements, without affecting the overall scalability of the Synapse protocol itself.

Interconnecting existing overlays made up of, so called, "blind" peers that are not aware of any additional parameters (the Synapse black box) seems to be a natural Synapse evolution and it constitutes a problem worth investigating. The assumption is that an overlay can be populated by blind peers (e.g. nodes previously in place) and synapses at the same time. Both interact in the same way in the overlay and exchange the same messages. Moreover, those synapses can be members of several overlays independently (thus being able to replicate a request from one overlay to another) and can communicate with each other exclusively through a dedicated overlay *Control Network*. The Control Network is basically a set of DHTs allowing each synapse to share routing information with other synapses without being aware of the routing of the undergoing message. So far the DHTs implemented are the following: (i) a Key table, responsible for storing unhashed keys circulating in the underlying overlays, every synapse accessing this table can easily retrieve the unhashed value of a key by using only the information it is aware of; (ii) a Replication table, in which is stored the number of times the key should be replicated across all of the the overlays; (iii) a Cache table, used to implement the replication of `GET` requests, and cache multiple responses and control the flooding of foreign networks.

## 3 Description of the Synapse Protocol Using the ASM Formalism

We assume that the reader is familiar with the semantics of the ASM. As part of this paper, we provide a brief overview of ASM in Appendix A.

The paper [9] describes the DHT-based protocol Chord in the setting of ASM, and gives proof of the conditions under which a system maintained by the Chord protocol forms stable and correct structure and distributes the keys over the nodes. As the Synapse protocol is not fully exhaustive [5] and its underlying structure depends on the protocols of all overlay networks participating in the system, in this section we restrict ourselves to extending [9] by presenting only a specification of the Synapse protocol in ASM.

Specification of the Synapse protocol that is given here follows pseudo-code given in [5].

Let $K, J, N$ and $M$ be three positive integers. We introduce the following disjoint universes:

- the set $Network = \{net_1, \ldots, net_N\}$ denotes all of the overlay networks present in the given system,
- the set $Hash = \{hash_1, \ldots, hash_N\}$ denotes hash functions for each of the overlay networks,
- the set $Node = \{node_1, \ldots, node_M\}$ represents the set of all of the nodes participating in the given system,
- the set $Key = \{key_1, \ldots, key_K\}$ denotes identifiers of objects that might be stored in the considered system, and the set $Value = \{value_1, \ldots, value_K\}$ represents the values of those $K$ objects,
- the set $Query = \{query_1, \ldots, query_J\}$ denotes all possible queries in the given system,
- the set $Action = \{join, leave, syn\_get, syn\_put\}$ represents the possible actions of a synapse node.

Each network in $Network$ is equipped with its own specific JOIN, LEAVE, PUT and GET rules. As these rules depend on the protocols in each of the intra-overlays, they cannot be specified formally at this point. Also, we introduce the following functions:

- $action : Node \rightarrow Action$, which saves current action of a node,
- $networkList : Node \rightarrow ListOfNetworks$, which maps every $node \in Node$ into a list of overlays in which that $node$ participates,
- $processed : Node \rightarrow ListOfQueries$, which registers already processed queries.
- $keyTable : Node \times Network \times Hash \rightarrow Key$, for connecting hashed and unhashed values of the keys for every overlay network,
- $cacheTable : Network \times Key \rightarrow ListOfValues$, used for caching already returned values,

The last two functions will be used only for the Black Box Synapse model.

### 3.1 Rules

During each execution of a *Synapse_agent Module*, which is defined in Section 3.2 below, the rules READMESSAGES, SYNGET and SYNPUT will be applied. The responsibility of the READMESSAGES rule is to process all of the messages sent to a particular node:

READMESSAGES=
Read Messages Dedicated To $Me$,
Change Local Variables If It Is Requested And Clear Processed Messages

If a synapse node applies one of the SYNPUT or SYNGET rules, the main operation is to invoke the PUT or GET rules of the underlying protocols, respectively. If a synapse node is queried by some other node, the same procedure extends the search space and, possibly, returns more answers. The White (Fig. 1) and Black Box (Fig. 2) models of these rules slightly differ. Namely, in the case of the White Box model, the functions $KeyTable$ and $CacheTable$ are not used because all of the nodes are aware of the changes made to the original protocol, while in the Black Box model the synapses need those functions in order to access and manipulate the unhashed keys. Here, we will give only the high level version of these rules. For the detailed version of the rules, we need to know all of the protocols that are used by all overlay networks and then, also, to change the rules of the basic protocols, i.e. to change all of the rules that are given in Appendix of [9].

### 3.2 Synapse module

The main module (Fig. 3) contains actions executed by every synapse node.

This module is executed in an infinite loop, with the appropriate rule(s) being applied in each of the iterations. With this module, we have formally defined the behavior of one Synapse node using ASM.

## 4 Exhaustiveness of the Synapse Protocol

In this section, we state the probability of the exhaustiveness of the Synapse protocol, under various assumptions.

We will be using "simple" probabilistic techniques to solve our problems. An alternative approach could involve the techniques developed in random graph theory, which might seem as a natural path to take, given the complicated inter-connected structure of the network. In the case of the Synapse protocol the configuration is not fully random. On the contrary, certain parts exhibit a high level of structure. If we would like to avoid this situation and try to model our problem using random multigraphs, then our focus would be to obtain probabilities of the existence of the paths of certain length, which is still an open problem.

First, we will use Lemmas 1 and 2, respectively, to give the probabilities of avoiding synapses in one overlay and avoiding all of the synapses which are members of another particular overlay. Then, we will give the probability of exhaustiveness of the Pure White Box model of the Synapse protocol, where there is no failure of the nodes, TTL is not limited, and synapse nodes are members of exactly two different overlay networks. Afterwards, we will examine how that probability changes if we allow failure of the nodes, limit the TTL, and allow for higher degrees of connectivity of the synapses. Also, the probability of exhaustiveness of the Black Box model of the Synapse protocol will be calculated.

To achieve statistical significance for the experiments and simulations performed in this Section, each configuration of the experiment or simulation was repeated between 1000 and 2000 times, depending on the experiment or simulation. Where the settings of the experiment and simulation corresponds the setting of a theorem we will use the results of obtained in [5].

**Lemma 1** *Let there be $b$ nodes in the overlay, where $w$ of them are not synapses, while the rest of them are. If the search procedure were to contact up to $l$ nodes (with uniform probability of choosing a number from $\{1,\ldots,l\}$), the probability of contacting no synapses is equal to:*

$$P^{(1)}_{w,b,l} = \frac{1}{l}\sum_{m=1}^{l}\frac{\binom{w}{m}}{\binom{b}{m}}.$$

*Proof* The probability of not contacting any of the synapses out of the $m$ nodes that have been contacted is equal to:

$$P_m = \frac{\binom{w}{m}}{\binom{b}{m}},$$

because $\binom{n}{k}$ is the number of combinations in which we could contact $k$ nodes out of the possible $n$. Since we can choose uniformly the number from the set $\{1,\ldots,l\}$, choose the number of nodes to be contacted, the final probability that we are looking for is equal to:

$$P^{(1)}_{w,b,l} = \frac{1}{l}\sum_{m=1}^{l} P_m = \frac{1}{l}\sum_{m=1}^{l}\frac{\binom{w}{m}}{\binom{b}{m}}.$$

□

**Lemma 2** *Let the system contain a number of overlays, and let $M_0$ and $M_1$ be two of these overlays. Let $M_0$ contain $b = w + r + g$ nodes, where $w$, $r$ and $g$ are the number of nodes that are not synapses, the number of synapses towards the overlay $M_1$, and the number of synapses to the remaining overlays in the system, respectively. If the search procedure were to contact up to $l$ nodes (with uniform probability of choosing the number from $\{1,\ldots,l\}$), the probability of contacting no synapses to $M_1$, if we know that at least one synapse has been contacted, is:*

$$P^{(2)}_{w,r,g,l} = \frac{1}{l}\sum_{m=1}^{l}\frac{\sum_{i=1}^{m}\binom{g}{i}\binom{w}{m-i}}{\sum_{i=1}^{m}\binom{g+r}{i}\binom{w}{m-i}}.$$

```
SynPut=
forall net with net ∈ networkList(Me)
  Invoke Put Of Network net To Store ⟨key, value⟩

SynGet=
if query ∉ processed(Me) and ttl > 0                    Check if the query is already
                                                        processed or TTL is reached
  forall net with net ∈ networkList(Me)
    par
      Invoke Get Of Network net To Find key
        with Reduces ttl
      processed(Me).add(query)
    endpar
```

Fig. 1: White Box Synapse Get and Put in ASM

```
SynPut=
seq
  Get Unhashed Value of key From keyTable
  forall net with net ∈ networkList(Me)
    Invoke Put Of Network net To Store ⟨key, value⟩
endseq

SynGet=
seq
  Invoke Get In Original Network
  Get Unhashed Value of key From keyTable
  if query ∉ processed(Me) and ttl > 0                  Check if the query is already
                                                        processed or TTL is reached
    par
      Get Results From cacheTable                       Check if a similar query
                                                        got the result
      forall net with net ∈ networkList(Me)
        par
          Invoke Get Of Network net To Find key
            with Reduces ttl
          processed(Me).add(query)
        endpar
    endpar
  Add Result To cacheTable                              Store the result for future queries
endseq
```

Fig. 2: Black Box Synapse Get and Put in ASM

*Proof* If $m$ nodes have been contacted, the probability of not contacting any of the synapses leading to $M_1$, if at least one synapse has been contacted, is:

$$P_m = \frac{\sum_{i=1}^{m} \binom{g}{i}\binom{w}{m-i}}{\sum_{i=1}^{m} \binom{g+r}{i}\binom{w}{m-i}}.$$

Similarly to Lemma 1, as we can uniformly, from the set $\{1, \ldots, l\}$, choose the number of nodes to be contacted, the final probability that we are looking for is equal to:

$$P^{(2)}_{w,r,g,l} = \frac{1}{l}\sum_{m=1}^{l} P_m = \frac{1}{l}\sum_{m=1}^{l} \frac{\sum_{i=1}^{m} \binom{g}{i}\binom{w}{m-i}}{\sum_{i=1}^{m} \binom{g+r}{i}\binom{w}{m-i}}.$$

□

### 4.1 Exhaustiveness of White Box Synapse

Hereinafter, we will assume that the complexity of the search procedure for any intra-/control overlay is $log_2(n)$. We will also use the following notation:

- $N, (N \geqslant 2)$ - total number of intra-overlays,
- $n$ - number of nodes per intra-overlay,
- $s$ - for the White Box model - percentage of nodes, that have become synapses; for the Black Box model - percentage of synapses with respect to all of the nodes in the system,
- $p_f$ - probability for a node/synapse to fail,
- $c$ - number of connections per synapse,
- $F$ - the event that the given key has been found,

```
seq
  READMESSAGES                                                    Process messages
  Choose An Action                                                Choose next action
  par
    if action(Me) = join
      seq
        Choose net To Join
        par
          JOIN network net                                        Invoke JOIN of net
          networkList(Me).add(net)                                Add net to local list
        endpar
      endseq
    endif
    if action(Me) = leave then
      seq
        Choose net To Leave
        par
          LEAVE network net                                       Invoke LEAVE of net
          networkList(Me).remove(net)                             Remove net from local list
        endpar
      endseq
    endif
    if action(Me) = syn_put then
      SYNPUT                                                      Invoke local SYNPUT
    endif
    if action(Me) = syn_get then
      SYNGET                                                      Invoke local SYNGET
    endif
  endpar
endseq
```

Fig. 3: Synapse module

- $KO$ - the event that the given key is stored at the same intra-overlay as the starting node,
- $S$ - the event that a synapse has been contacted,
- $D$ - the event that a query passed a maximum of $D$ intra-overlays,
- $B^c$ - the complementary event of some event $B$.

*Pure White Box Synapse* In this scenario, the nodes do not fail, TTL is not limited, and synapse nodes are members of exactly two different overlay networks.

**Theorem 1 (Pure White Box Satisfaction Ratio)** *The probability for a node to get a value for a given key stored in the system, where all the synapses are connecting exactly two overlays, is:*

$$P(F) = 1 - \frac{N-1}{N}\left(P^{(1)}_{(1-s)n,(1+s)n,l} + \left(1 - P^{(1)}_{(1-s)n,(1+s)n,l}\right)\left(P^{(2)}_{(1-s)n,\frac{2ns}{N-1},2ns\frac{N-2}{N-1},l}\right)^{N-1}\right),$$

*where* $l = \lfloor log_2((1+s)n) \rfloor$.

*Proof* The probability $P(F)$ to find the given key is equal to:

$$P(F) = 1 - P(F^c).$$

There are two possibilities: the given key and the starting node are or are not stored in the same overlay. Therefore, using the formula of total probability, we have that:

$$P(F^c) = P(KO)P(F^c|KO) + P(KO^c)P(F^c|KO^c).$$

Due to the property of sub-overlays that if a key is stored in a particular overlay it will always be found, we have that $P(F^c|KO) = 0$, so:

$$P(F^c) = P(KO^c)P(F^c|KO^c).$$

Next, we have that the probability that the given key and the starting node are not in the same overlay is equal to:

$$P(KO^c) = \frac{N-1}{N}.$$

There are two cases in which we do not get the key that is stored in a different overlay from the starting node. In the first case, none of the synapses have been reached during the search procedure at the starting overlay. In the second case, at least one of the synapses was asked but anyway the key was not found (because the query didn't reach the overlay where the key is stored). This is reflected by:

$$P(F^c|KO^c) = P(S^c)P((F^c|KO^c)|S^c) + P(S)P((F^c|KO^c)|S),$$

where $P((F^c|KO^c)|S^c)$ is equal to 1. Finally, we get:

$$P(F) = 1 - \frac{N-1}{N}(P(S^c) + P(S)P((F^c|KO^c)|S)). \quad (1)$$

From Lemma 1, we can obtain $P(S^c)$. We can consider the situation that the overlay contains $(1+s)n$ nodes while $2sn$ of them are synapses, from which we get:

$$P(S^c) = P^{(1)}_{(1-s)n,(1+s)n,l}.$$

Also, we have that:

$$P(S) = 1 - P(S^c).$$

Similarly, from Lemma 2 we can get $P((F^c|KO^c)|S))$. This time we can consider $N-1$ overlays with $(1+s)n$ nodes in total, while $(1-s)n$ are not synapses, and $\frac{2ns}{N-1}$ are those synapses which lead to the particular overlay. We have to adopt this for all $N-1$ overlays that are not in the starting network, yielding:

$$P((F^c|KO^c)|S)) = P^{(2)}_{(1-s)n,\frac{2ns}{N-1},2ns\frac{N-2}{N-1},l}{}^{N-1}.$$

Now, we have all of the components required for equation (1):

$$P(F) = 1 - \frac{N-1}{N}\Bigg(P^{(1)}_{(1-s)n,(1+s)n,l} + \left(1 - P^{(1)}_{(1-s)n,(1+s)n,l}\right)\left(P^{(2)}_{(1-s)n,\frac{2ns}{N-1},2ns\frac{N-2}{N-1},l}\right)^{N-1}\Bigg),$$

□

In Figure 4, we present the results of the deployment of openSynapse and JSynapse, the applications of the Synapse protocol developed and tested for the purposes of [5], as well as the graph constructed from the result of Theorem 1. The lines represent various experiments where the given number of nodes was uniformly distributed over the given number of overlay networks, as described in the corresponding legends. The percentage of the nodes that have become synapses is given on the x-axis. We can see from the graphs that there exists a substantial correspondence between the theoretically predicted results and those obtained through experimentation.

*White Box Synapse with Node Failure* In this scenario, with respect to the previous one, we allow the possibility for a node to fail.

**Lemma 3** *The expected number of nodes per overlay is:*

$$\mathbb{E}(n) = (1-p_f)n$$

If we were to allow node failures in Theorem 1, the number of nodes we have per overlay would be $\mathbb{E}(n)$ rather then $n$. Also, we have that the probability that a given key was not stored on some of the nodes that have failed is $1-p_f$. With this, we obtain the following result:

**Theorem 2** *The probability for a node to get a value for a given key stored in the system is:*

$$P(F) = (1-p_f)\Bigg(1-\frac{N-1}{N}\Bigg(P^{(1)}_{(1-s)(1-p_f)n,(1+s)(1-p_f)n,l} + \left(1 - P^{(1)}_{(1-s)(1-p_f)n,(1+s)(1-p_f)n,l}\right)\left(P^{(2)}_{(1-s)n,\frac{2(1-p_f)ns}{N-1},2(1-p_f)ns\frac{N-2}{N-1},l}\right)^{N-1}\Bigg)\Bigg),$$

*where* $l = \lfloor log_2((1+s)n) \rfloor$.

*White Box Synapse with Multiple Connectivity of Synapses* In this scenario, we allow synapses to connect more than two overlays at the same time. Also, we do not take into consideration node failures.

**Theorem 3** *The probability for a node to get a value for a given key stored in the system, where all the synapses are connecting exactly* $c$ *overlays, is:*

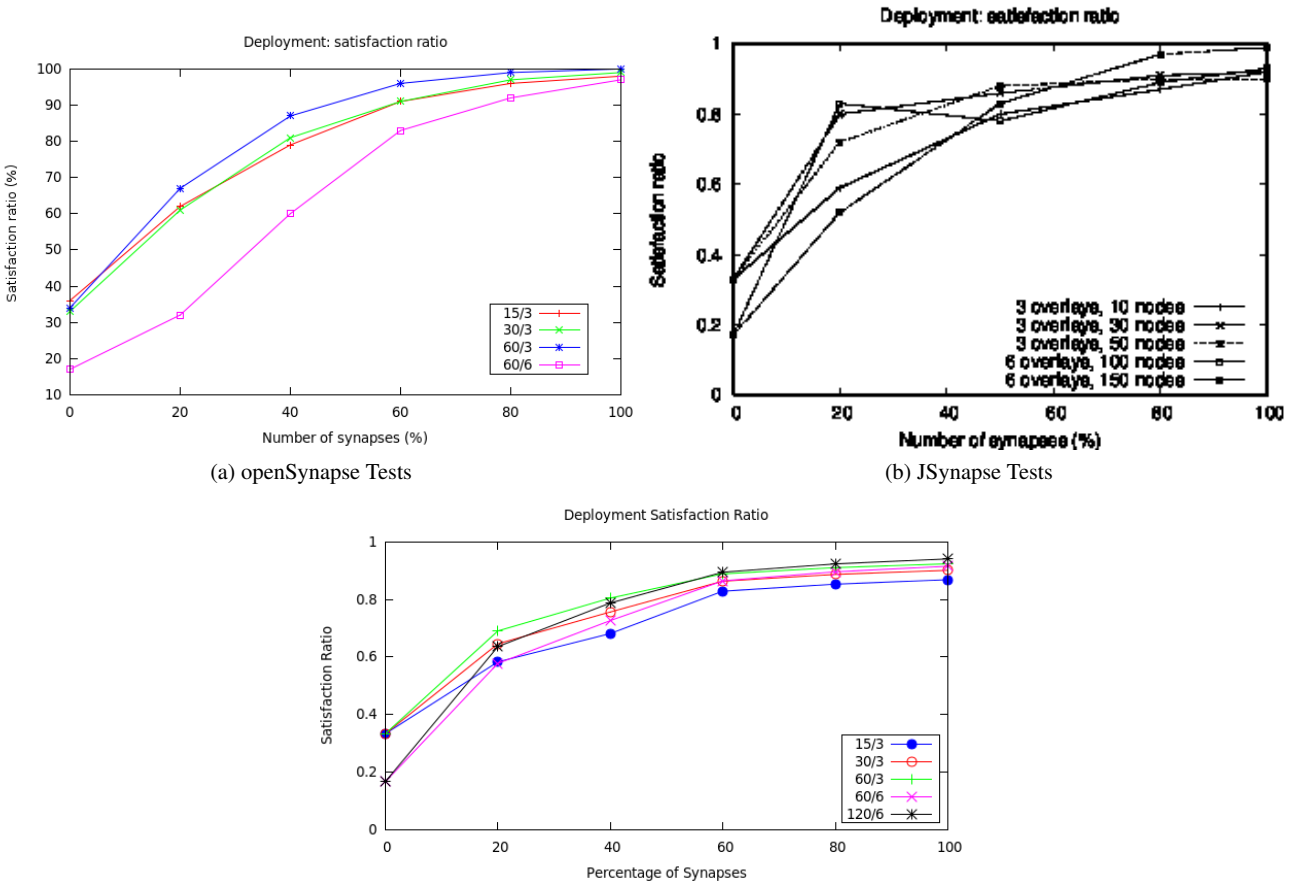
$$P(F) = 1 - \frac{N-1}{N}\Bigg(P^{(1)}_{(1-s)n,(1+(c-1)s)n,l} + \left(1 - P^{(1)}_{(1-s)n,(1+(c-1)s)n,l}\right)P^{(2)}_{(1-s)n,\frac{cns}{N-1},cns\frac{N-2}{N-1},l}{}^{N-1}\Bigg),$$

*where* $l = \lfloor log_2((1+(c-1)s)n) \rfloor$.

*Proof* The total number of nodes in Lemmas 1 and 2 increases with every new connection of a new synapse, so we can consider every new connection as a new node of the underlying overlay network. □

In Figure 5, we illustrate the situation where 10000 nodes are uniformly distributed over 20 overlay networks. The lines tell us the percentage of the nodes which are transformed to synapses, while on the x-axis we present the degree of connectivity of the synapses. Again, we can see that theoretical predictions correspond to the obtained experimental results.

*White Box with TTL* In this scenario, we allow for a time-to-live (TTL) stopping criterion for the issued request. If we consider TTL as the number of overlays that can be reached during one query, then we have the following theorem:




(a) openSynapse Tests

(b) JSynapse Tests

(c) Theoretical Calculations

Fig. 4: Satisfaction Ratio - Experiments and Theory

**Theorem 4** *The probability for a node to get a value for a given key stored in the system is:*

$$P(F) = 1 - \frac{N-1}{N}\left(\frac{D}{N}\left(P^{(1)}_{(1-s)n,(1+s)n,l} + \left(1 - P^{(1)}_{(1-s)n,(1+s)n,l}\right)\left(P^{(1)}_{(1-s)n,(1+s)n,l} + \left(1 - P^{(1)}_{(1-s)n,(1+s)n,l}\right)\left(P^{(2)}_{(1-s)n,\frac{2ns}{D-1},2ns\frac{D-2}{D-1},l}\right)^{D-1}\right)\right) + \frac{N-D}{N}\right)$$

*where $l = \lfloor log_2((1+s)n) \rfloor$, and $D$ is the maximum allowed number of overlays for a query to pass.*

*Proof* Similarly to the proof of Theorem 1, we can get that:

$$P(F) = 1 - \frac{N-1}{N}P(F^c|KO^c),$$

where:

$$P(F^c|KO^c) = \\ = P(D)P((F^c|KO^c)|D) + P(D^c)P((F^c|KO^c)|D^c).$$

Next, since the probability $P((F^c|KO^c)|D^c)$ of not finding the given key is 1, and $P(D) = \frac{D}{N}$, we have that:

$$P(F^c|KO^c) = \frac{D}{N}P((F^c|KO^c)|D) + \frac{N-D}{N}.$$

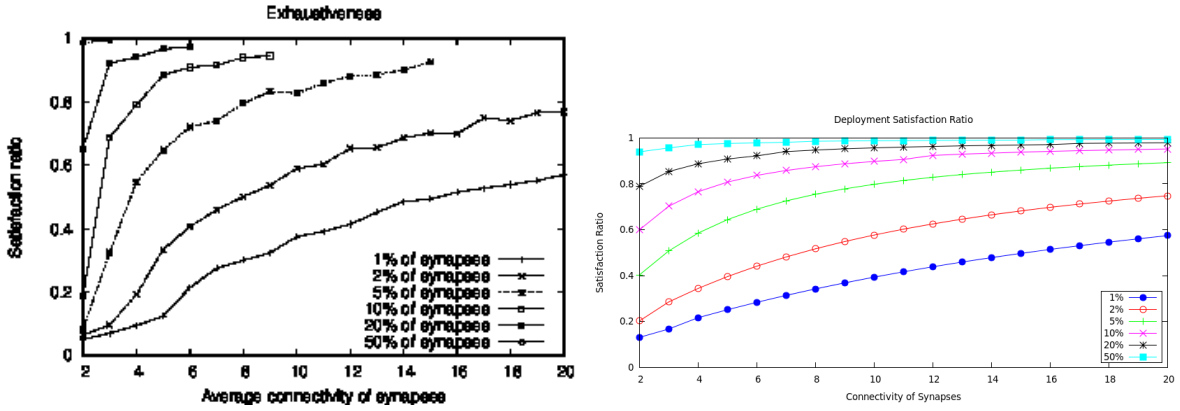
Like in the proof of Theorem 1, the event $(F^c|KO^c)|D$ can be divided into whether the synapse in the starting network has or has not been contacted, but this time taking into account a system containing $D$ overlays:

$$P(F^c|KO^c)|D) = P^{(1)}_{(1-s)n,(1+s)n,l} + \\ (1 - P^{(1)}_{(1-s)n,(1+s)n,l})P((F^c|KO^c)|D)|S^c),$$

and

$$P((F^c|KO^c)|D)|S^c) = P^{(1)}_{(1-s)n,(1+s)n,l} + \\ \left(1 - P^{(1)}_{(1-s)n,(1+s)n,l}\right)\left(P^{(2)}_{(1-s)n,\frac{2ns}{D-1},2ns\frac{D-2}{D-1},l}\right)^{D-1},$$

which completes our equation. □




(a) py-synapse Simulations (b) Theoretical Calculations

Fig. 5: Connectivity of Synapses - Simulations and Theory

In Figure 6, we examine a system of 1000 nodes uniformly distributed over 10 overlay networks. On the graphs the scenarios with different percentages of nodes that have become synapses and TTL are represented. Again, one can notice a clear correspondence between the theory and the experiments.

### 4.2 Exhaustiveness of Black Box Synapse model

**Theorem 5 (Pure Black Box Satisfaction Ration)** *The probability for a node to get a value for a given key stored in the system is:*

$$P(F) = 1 - \frac{N-1}{N}\left(P^{(1)}_{(1-s)n,n,l} + \left(1 - P^{(1)}_{(1-s)n,n,l}\right)\right) P^{(2)}_{(1-s)n,\frac{sn}{N-1},sn\frac{N-2}{N-1},l}{}^{N-1} P^{(1)}_{sn(N-1),snN,L}.$$

*where* $l = \lfloor log_2(n) \rfloor$ *and* $L = \lfloor log_2(snN) \rfloor$.

*Proof* Similarly as in the proof of Theorem 1, we obtain:

$$P(F) = 1 - \frac{N-1}{N}(P(S^c) + P(S)P((F^c|KO^c)|S)). \quad (2)$$

We have that

$$P(S^c) = P^{(1)}_{(1-s)n,n,l}.$$

and also that $P(S) = 1 - P(S^c)$. This time, besides the $N$ sub-overlay networks, we have one more control network consisting of all of the synapses in the system. Therefore, we have that:

$$P((F^c|KO^c)|S)) = \left(P^{(2)}_{(1-s)n,\frac{sn}{N-1},sn\frac{N-2}{N-1},l}\right)^{N-1} P^{(1)}_{sn(N-1),snN,L},$$

which completes equation 2. □

Similarly, like in the previous section, we can extend Pure Black Box Model and obtained formula by allowing synapse failures or their multiple connectivity and introducing TTL. In Figure 7, we show the results of the deployment of jSynapse, as well as the graph illustrating the result of Theorem 5. The lines represent various experiments where the given number of nodes was uniformly distributed over the given number of overlay networks, while the total number of synapses per overlay is given on the x-axis. Just as in the previous three cases, we can notice a clear correspondence between the graphs.

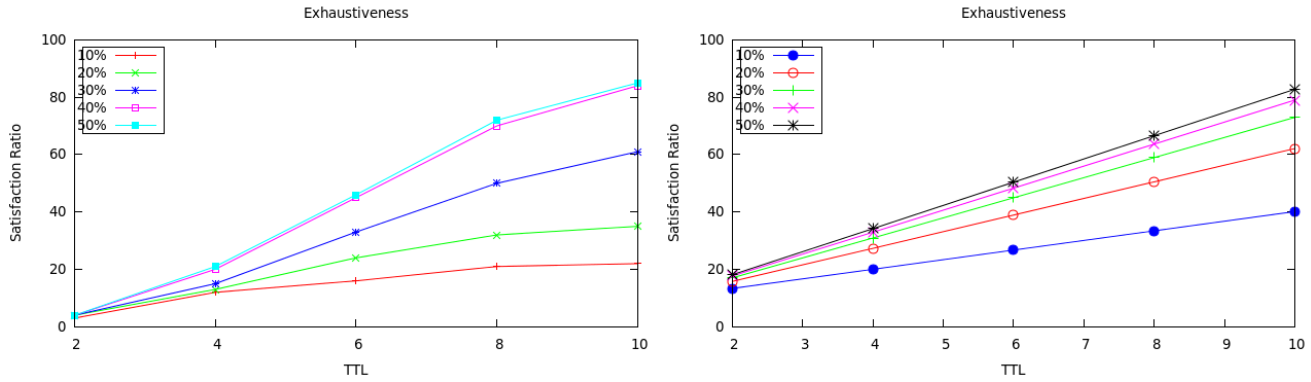
## 5 Improvements of Synapse

The implementation of the Synapse protocol implemented in previous sections will be referred to as "opportunistic routing", i.e., a request could transit to a other intra-overlay only if a synapse node was reached during the routing request in the original intra-overlay.

Furthermore, we shall remind, that in order to be reliably performed, the inter-routing operation requires additional data (amongst which, the non-hashed version of the requested key) to be exchanged between a node and a synapse. This exchange happens according to two different mechanisms, depending on the "openness" of the considered system. Similarly like in the previous sections two models of the protocol can be considered: the **White** and **Black** box.

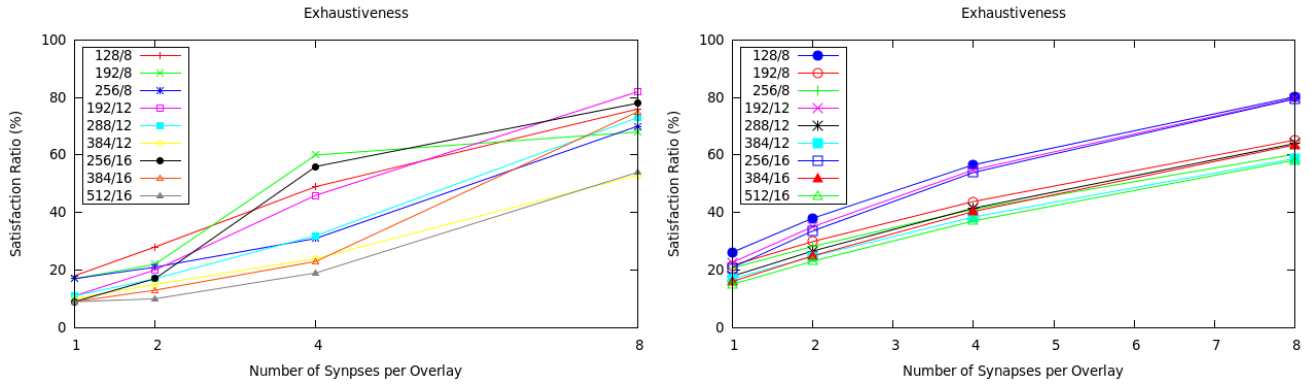
In this section, we present a set of several improvements over the original system in order to provide, rather than a simple routing protocol, a more general framework to allow the design of routing schemes and applications based on multiple overlay networks interconnected via co-located nodes. In particular, this new system, presented in [6] and [7], differs from the one in [5] in the following aspects:

– instead of embedding sensitive inter-routing data in the overlay messages, or sharing it via a DHT, the data is




(a) py-synapse Simulations (b) Theoretical Calculations

Fig. 6: TTL - Simulations and Theory




(a) jSynapse simulations (b) Theoretical Calculations

Fig. 7: Black Box Synapse - Simulations and Theory

now exchanged only between the requesting peer and a gateway node via a dedicated encrypted message;
- peers can discover new gateway nodes, thus building an unstructured overlay neighborhood on top of the structured one, by different mechanisms, namely: overlay message analysis, explicit notifications or peer exchange;
- rather than relying on the sole opportunistic routing, the system can now implement different routing strategies that send inter-routing requests to gateway nodes in a peer's neightborhood at the same time as requests to the structured overlays it is connected to.

Thanks to these major improvements, we now have an entire new general architecture capable of orchestrating multiple overlay networks as well as transparently interconnecting existing overlays, with the possibility of designing new inter-routing schemes that may vary depending on the application or the networking conditions. We can choose, for example, to explicitly select an overlay to route a request into according to ustructured cryteria, (flooding, random walk, etc.), or we can decide to arrange the IDs of each overlay according to a structured model and perform inter-overlay routing following a structured mechanism, being as such able to emulate the behavior of certain hierarchical overlays. Furthermore, we can still rely on the original mechanism of routing if a synapse node is touched (opportunistic routing), thus maintaining the same desirable properties of the first protocol implementation.

Node discovery can also be performed in an opportunistic way, by embedding additional information in the overlays messages, or independently via a peer exchange mechanism. The latter allows for a Synapse node to be completely independent from the underlying overlay protocol, thus giving the possibility of having existing overlay networks interact with each other without breaking network compatibility, by using the same protocol that we would use for a white box scenario. As such, this new implementation aims to be a generalization and an extension of the original Synapse protocol to multiple cooperation scenarios.

**Theorem 6 (Improved Synapse)** *The probability $P(F)$ for a node to get a value for a given key stored in the system is equal to:*

$$P(F) = 1 - (1-s)\frac{1}{l}\sum_{m=1}^{l}\frac{\binom{(1-s)n}{m}}{\binom{n}{m}}.$$

*Proof* The probability to get an answer is equal to:

$$P(F) = P(S)P(F|S) + P(S^c)P(F|S^s),$$

where $P(S)$ denotes the probability of a event that a query starts from a synapse, and $P(F|S)$ denotes the probability to get an answer under that condition. Next, the probability $P(F|S)$ is 1 and the probability to pick a synapse is equal to $s$, so:

$$P(F) = s + (1-s)\left(1 - \frac{1}{l}\sum_{m=1}^{l}\frac{\binom{(1-s)n}{m}}{\binom{n}{m}}\right).$$

Finally, we have that:

$$P(F) = 1 - (1-s)\frac{1}{l}\sum_{m=1}^{l}\frac{\binom{(1-s)n}{m}}{\binom{n}{m}}.$$

□

Figure 8 compares the results of Theorem 1 and 6 in the situation when the system consists of 2000 nodes uniformly distributed on the number of overlay networks given on x-axis with different connectivity of the synapses, showing the superiority of improved Synapse over the original one.

## 6 Conclusions

In this paper, using the formalism of ASM, we have provided the specification of the Synapse protocol. We have given probabilistic assessments of the exhaustiveness of this protocol under a variety of scenarios. We can conclude that the exhaustiveness equations that have been proven are in strong correspondence with the results obtained by running the appropriate simulations and experiments. These equations have shown us that good exhaustiveness can be reached with a relatively small percentage of strategically positioned synapse nodes. It is always better to have a higher degree of connectivity of the synapses and an unlimited TTL, but even with relatively small numbers, reasonably good exhaustiveness can be achieved.

Possible directions for further work include the application of similar techniques for solving some open problems in the field of random multigraphs, especially when some parts of those graphs are not fully random but very-well structured.

## References


1. E. Börger, R. Stärk. *Abstract State Machines A Method for High-Level System Design and Analysis.*, Springer-Verlag, 2003.
2. V. Ciancaglini, L. Liquori, L. Vanni. *CarPal: interconnecting overlay networks for a community-driven shared mobility.*, in *Trustworthy Global Computing 2010.*
3. Y. Gurevich. *Evolving Algebras 1993: Lipari Guide.* In *Specification and Validation Methods*, Oxford University Press, pages 9–36, 1995.
4. Y. Gurevich. *Sequential Abstract State Machines capture Sequential Algorithms.* In *ACM Transactions on Computational Logic Volume 1, Number 1*, pages 77–111, 2000.
5. L. Liquori, C. Tedeschi, L. Vanni, F. Bongiovanni, V. Ciancaglini and B. Marinković. *Synapse: A Scalable Protocol for Interconnecting Heterogeneous Overlay Networks*. In Networking 2010, Lecture Notes in Computer Science, vol. 6091 (p. 410), pages 67–82, 2010.
6. V. Ciancaglini, G.N. Hoang and L. Liquori. *Towards a Common Architecture to Interconnect Heterogeneous Overlay Networks*. In ICPADS 2011, IEEE, pages 817 – 822, 2011.
7. V. Ciancaglini, G.N. Hoang, P. Maksimović and L. Liquori. *An Extension and Cooperation Mechanism for Heterogeneous Overlay Networks*. In Networking 2012, Lecture Notes in Computer Science, vol. 7291, pages 10 – 18, 2012.
8. B. Marinković, L. Liquori, V. Ciancaglini and Z. Ognjanović. *A Distributed Catalog for Digitized Cultural Heritage*. In *ICT Innovations 2010*, CCIS 83, pages 176 – 186, 2011.
9. B. Marinković, P. Glavan and Z. Ognjanović. *Formal Description of the Chord Protocol using ASM*. At arXiv:1208.0712v1.


## A Abstract State Machines

We assume that the reader is familiar with the semantics of the Abstract State Machine defined in [1,3,4], and we quote here only the essential definitions.

A Gurevich's Abstract State Machine $\mathcal{A}$ is defined by a program *Prog* - consisting of a finite number of *transition rules*, at most countable set of states and initial states. $\mathcal{A}$ models the operational behavior of a real dynamic system $\mathcal{S}$ in terms of evolution of states.

A state $S$ is a first-order structure over a fixed signature (which is also the signature of $\mathcal{A}$), representing the instantaneous configuration of $\mathcal{S}$. The value of a term $t$ at $\mathcal{S}$ is denoted by $[t]_S$. The basic transition rule is the following function update

$$f(t_1, \ldots, t_n) := t$$

where $f$ is an arbitrary $n$-ary function and $t_1, \ldots, t_n, t$ are first-order terms. To fire this rule in a state $S$ evaluate all terms $t_1, \ldots, t_n, t$ at $S$ and update the function $f$ to $[t]_S$ on parameters $[t_1]_S, \ldots, [t_n]_S$. This produces another state $S'$ which differs from $S$ only in the new interpretation of the function $f$ (since states represent memory, function update represents change in the content of one memory location).

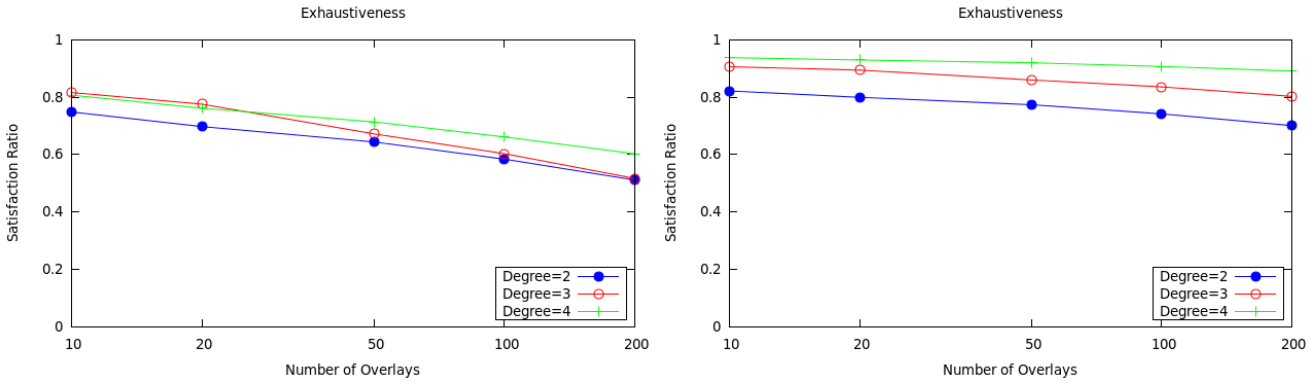
Additionally, we have the following transition rules.

The *conditional constructor* produces "guarded" transition rules of the form:

```
if g then
  R1
else
  R2
endif
```

where $g$ is a ground term (the guard of the rule) and $R_1$, $R_2$ are transition rules. To fire that new rule in a state $S$ evaluate the guard; if it is true, then execute $R_1$, otherwise execute $R_2$. The `else` part may be omitted.

The *seq constructor* produces transition rules of the form:




(a) Pure White Box Synapse (b) Improved Synapse

Fig. 8: Comparison of the Expected Results between Pure White Box Synapse and Improved Synapse

```
seq
  R_1
  ...
  R_n
endseq
```

to apply $R_1, \ldots, R_n$ sequentially. Note that the *seq*-constructor is originally defined without the `endseq`-line, but we add it to improve readability.

The *par constructor* produces transition rules of the form:

```
par
  R_1
  ...
  R_n
endpar
```

to apply $R_1, \ldots, R_n$ simultaneously, when possible, otherwise do nothing.

If $U$ is a universe name, $v$ is a variable, $g(v)$ is a Boolean term and $R$ is a rule then the following expression (*choose constructor*) is a rule with the main existential variable $v$ that ranges over $U$ and body $R$:

```
choose v in U satisfying g(v)
  R
endchoose
```

If there is an element $a \in U$ such that condition $g(a)$ is true, fire rule $R$ (with $a$ substituted for $v$), otherwise do nothing.

To express the simultaneous execution of a rule $R$ for each $x$ satisfying a given condition $\varphi$ (*forall constructor*):

```
forall x with φ do
  R
endforall
```

A run/computation of $\mathcal{A}$ is a finite or infinite sequence $S_0; S_1; \ldots$ where $S_0$ is an initial state and every $S_{i+1}$ is obtained from $S_i$ executing a transition rule.

In general runs may be affected by the environment. Environment manifests itself via so-called external functions. Every external function can be understand as a (dynamic) oracle. The ASM provides the arguments and the oracle gives the result.

In a distributed Gurevich's Abstract State Machine $\mathcal{A}$ multiple autonomous agents cooperatively model a concurrent computation of $\mathcal{S}$. Each agent $a$ executes its own single-agent program $Prog(a)$ as specified by the module associated with $a$ by the function $Mod$. More precisely, an agent $a$ has a partial view $View(a; S)$ of a given global state $S$ as defined by its sub-vocabulary $Fun(a)$ (i.e. the function names occurring in $Prog(a)$) and it can make a move at $S$ by firing $Prog(a)$ at $View(a; S)$ and changing $S$ accordingly. The underlying semantic model ensures that the order in which the agents of $\mathcal{A}$ perform their actions is always such that no conflicts between the update sets computed for distinct agents can arise. The global program $Prog$ is the union of all single-agent programs. Nullary function $Me$, that allows an agent to identify itself among other agents, is interpreted as $a$ for each agent $a$, and does not belong to $Fun(a)$ for any agent $a$. It cannot be the subject of an update instruction and is used to parameterize the agent's specific functions. A sequential run of a distributed Gurevich's Abstract State machine $\mathcal{A}$ is a (finite or infinite) sequence $S_0; S_1; \ldots; S_n; \ldots$ of states of $\mathcal{A}$, where $S_0$ is an initial state and every $S_{n+1}$ is obtained from $S_n$ by executing a move of an agent. The partially ordered run, defined in [3], is the most general definition of runs for a distributed ASM. In order to prove properties on a partially ordered run, the attention may be restricted to a linearization of it, which is, in turn, a sequential run (see [3] for more explanations). In the rest of the paper we consider only *regular runs* in which a state is global and moves of agents are atomic.